DAMOCO: MATLAB toolbox for multivariate data analysis, based on coupled oscillators approach Version 1.0

Björn Kralemann[1], Michael Rosenblum[2], Arkady Pikovsky[2]

[1] Institut für Pädagogik, Christian-Albrechts-Universität zu Kiel
Olshausenstr. 75, 24118 Kiel, Germany

[2] Institute of Physics and Astronomy, Potsdam University
Karl-Liebknecht-Str. 24-25, 14476 Potsdam, Germany

January 17, 2011

Abstract

This manual describes the collection of MATLAB programs for multivariate data analysis, based on modeling the signal sources by coupled oscillators. The programs and the manual can be downloaded from `www.agnld.uni-potsdam.de/~mros/damoco.html`.

1 Introduction

DAMOCO means **D**ata **A**nalysis with **M**odels **O**f **C**oupled **O**scillators. This MATLAB toolbox is designed to analyze multivariate data be means of the coupled oscillators approach [1, 2, 3, 4, 5]. The functions presented here implement algorithms, developed in [3, 6, 7]. The main ideas and assumptions of the approach are briefly summarized below.

- The object of our analysis are active oscillatory systems, i.e. self-sustained oscillators. In biology such systems are called endogenous. Our approach works for noisy limit cycle oscillators or weakly chaotic oscillators.
- We assume that signals we analyze are generated by such systems. This assumption is crucial for the interpretation of results.
- The signals should be appropriate for phase estimation (i.e. they should be narrow-band signals) or should be made appropriate by a corresponding preprocessing. We do not provide the functions for preprocessing because it heavily depends on the particular application.
- The key issue of our approach is distinction between **phases** and **protophases**.
- According to the theory [8, 3], phase of an autonomous endogenous (self-sustained) oscillator grows linearly with time:
$$\dot{\phi} = \omega_0 , \qquad \phi = \omega_0 t + \phi_0 ,$$
where ω_0 is the natural frequency of the isolated system.
- In the practical data analysis phase is typically estimated via construction of a two-dimensional embedding. The first coordinate in this embedding is the time series $x(t)$ itself, while the second one can be chosen in different ways. Often this second coordinate is the Hilbert transform $x_H(t)$ of

the original signal, or its time derivative $\dot{x}$, or the result of the complex wavelet transform. In case of numerical simulation of an oscillating system, for the second coordinate of the embedding one can choose any other coordinate of the dynamical system. The phase estimate, or **protophase** is obtained as the angle in this embedding, i.e.

$$\theta = \arctan\left(\frac{x_H - y_0}{x - x_0}\right) , \qquad \text{or} \qquad \theta = \arctan\left(\frac{\dot{x} - y_0}{x - x_0}\right) \qquad \text{or}\ldots , \tag{1}$$

where x_0, y_0 are coordinates of the point, taken for the origin. The crucial issue is that, generally, a protophase θ obtained in this way, does not possess the property of the true phase and does not grow linearly in time. Generally, the instantaneous frequency is a function of the phase itself:

$$\dot{\theta} = \omega_0 + g(\theta) .$$

It is important, that the oscillation of the frequency, described by $g(\theta)$ does not have any physical meaning, but depends on the observable and embedding used, on the preprocessing, coordinates x_0, y_0, etc.

- In order to obtain results, independent of the details of measurement and preprocessing, we transform the protophases to true phases. This transformation has the form

  $$\frac{d\phi}{d\theta} = \sigma(\theta) ,$$

  where σ is the transformation function.

- True phases of two interacting oscillators obey the phase dynamics equations

  $$\begin{aligned} \dot{\phi}_1 &= \omega_1 + Q^{(1)}(\phi_1, \phi_2) , \\ \dot{\phi}_2 &= \omega_2 + Q^{(2)}(\phi_2, \phi_1) , \end{aligned} \tag{2}$$

  where $Q^{(1,2)}$ are the coupling functions. The protophases obey equations

  $$\begin{aligned} \dot{\theta}_1 &= \tilde{\omega}_1 + F^{(1)}(\theta_1, \theta_2) , \\ \dot{\theta}_2 &= \tilde{\omega}_2 + F^{(2)}(\theta_2, \theta_1) , \end{aligned} \tag{3}$$

  where the functions $F^{(1,2)}$, in contrast to functions $Q^{(1,2)}$, are non-universal. It means that $F^{(1,2)}$ depend on the observables and the embedding used for their calculation. In particular, functions $F^{(1,2)}$ typically strongly dependent on the own phase (i.e. $F^{(1)}$ strongly depends on θ_1, see, e.g. Fig. 2 in)`example2.pdf`, whereas this dependence does not reflect the true dynamics of the coupled oscillators but rather the choice of protophases. Our analysis is based on the reconstruction of the invariant functions $Q^{(1,2)}$. If these functions are known, then the interaction of oscillators can be characterized in an invariant way. **Note.** In many cases the functions of the toolbox compute the right hand side of Eqs. (2) or (3); therefore in the code the notations `f,q` often stay for $\omega + Q$ or $\omega + F$, especially in the intermediate computations. The final results provide separately frequencies ω and coupling functions F, Q.

1.1 List of functions

If you do not want to go into details, you can simply use the function `co_fbtransf1` for univariate analysis or functions `co_transf2`, `co_ittransf2` for bivariate analysis. If you want to optimize your particular application, you may use the building blocks of the algorithms, which are implemented as separate MATLAB functions.

All functions have a common name prefix `co_`, what stays for **c**oupled **o**scillators. The current version 1.0 of the toolbox contains the following files

1. General purpose functions.
   (a) `co_testproto:` Auxiliary function to test input data.
   (b) `co_hilbproto:` Computation of instantaneous protophase from a scalar time series, using the Hilbert transform.
   (c) `co_sync:` Computation of the $n : m$ synchronization index.
   (d) `co_maxsync:` Maximal $n : m$ synchronization index for a given range of n, m.
   (e) `co_dirin:` Directionality index from norms of the coupling functions.
   (f) `co_dirpar:` Directionality index from partial derivatives of the coupling functions.
2. Univariate transformation.
   (a) `co_fbtransf1:` Fourier based univariate transformation from protophase to phase.
3. Bivariate transformation.
   (a) `co_fbtransf2:` High-level function which performs two-dimensional protophase to phase transformation and computes the coupling functions, frequencies, directionality index. Fourier-based technique.
   (b) `co_ittransf2:` High-level function which performs two-dimensional protophase to phase transformation and computes the coupling functions, frequencies, directionality index. Iteration technique.
   (c) `co_fexp2:` Given two (proto)phases, the function yields two coupling functions via fitting a Fourier series.
   (d) `co_fexp1:` Similar to `co_fexp1`, but only one coupling function is computed.
   (e) `co_fbsolv:` Using the output of `co_fexp2`, this function computes the bivariate protophase to phase transformation functions $\sigma_{1,2}$.
   (f) `co_fbth2phi:` $\theta_{1,2} \to \phi_{1,2}$ transformation, using the output of `co_fbsolv`.
   (g) `co_fbnorm:` Norm of the coupling function, given by its Fourier coefficients.
   (h) `co_itersolv:` Using the output of `co_fexp2`, this function computes the bivariate transformation functions $\sigma_{1,2}$ by iteration technique; it also returns frequencies, true coupling functions, and their norms.
   (i) `co_gth2phi:` Using the output of `co_itersolv`, this function performs the bivariate $\theta_{1,2} \to \phi_{1,2}$ transformation.
4. Additional functions.
   (a) `co_plotcplf:` Plot of the coupling function.
   (b) `co_plot2cplf:` Plot of two coupling functions in the same window.
   (c) `co_plotcoef:` Plot of the Fourier coefficients of the coupling function.
   (d) `co_plot2coef:` Plot of the Fourier coefficients of two coupling functions in the same window.
   (e) `co_fbcfcor:` Correlation between two coupling functions (Fourier-based).
   (f) `co_cfcor:` Correlation of two coupling functions, given on a grid.

1.2 Conventions

The functions of the toolbox use the following conventions:

1. Input data are protophases computed in the $[0, 2\pi]$ interval. Phases, computed via a transformation are also by default not unwrapped, i.e. they are from 0 to 2π.

2. Protophases and phases are stored as column vectors.

3. If a function takes two (proto)phases as inputs, they should be column vectors of the same length.

4. The first argument of a coupling function is its own (proto)phase, i.e. the functions, stored as matrices, are $F^{(1)}(\phi_1, \phi_2)$ and $F^{(2)}(\phi_2, \phi_1)$.

5. Many toolbox functions operate on a grid, e.g. `co_fbtransf1` provides the transformation function $\sigma(\theta)$ computed on a grid of size N (typically denoted as `ngrid`.) This means that θ takes the discrete values
$$\theta_i = 0, \frac{2\pi}{N-1}, 2\frac{2\pi}{N-1}, \ldots, 2\pi \ , \quad i = 1, \ldots, N$$
and $\sigma_1 = \sigma_N$ due to periodicity.

6. Some functions optionally produce graphic output. These functions have input parameter `fignum`, which should be zero or positive integer. If `fignum=0`, then no plot is produced, otherwise the function opens a graphic window by command `figure(fignum);` and plot data in this window.

1.3 Using the toolbox

Using the toolbox is simple: (i) create an directory and put there all `.m` files; (ii) use the MATLAB command `pathtool` to make this directory known to MATLAB; (iii) use the toolbox functions like any other MATLAB function. So, e.g., type

```
>> help co_cplfct1
```

in the MATLAB command line to get help for the function `co_cplfct1.m`, or type

```
>> [phi,argsi,sigma]=co_fbtransf1(theta);
>> plot(argsi,sigma);
```

to transform the column vector of protophases `theta` into phases `phi` and to plot the transformation function `sigma`.

2 Description of the toolbox functions

Here we give a brief description of the toolbox functions. The form of the call, description of input and output parameters can be found in the function's help.

2.1 General purpose functions

2.1.1 Test of input data

The input to most of the functions are time series of protophases. The function `co_testproto` checks the input data for consistency and for accordance with the above formulated conventions. The input parameters are one or two protophases. The function returns 1, if data are correct, and zero, if something is wrong.

2.1.2 The Hilbert protophase

The function `co_hilbproto` computes the Hilbert protophase from a scalar time series. It allows to adjust the origin by setting its coordinate x_0, y_0 (see Eq. 1). In order to eliminate the boundary effects of the transformation, it cuts the parts of the signal at the beginning and at the end. It also gives a warning if the phase is ill-defined.

2.1.3 Synchronization index

The function `co_sync` computes the $n : m$ synchronization index, also known as the phase locking value:

$$\gamma_{n,m} = |\langle e^{i(n\theta_1 - m\theta_2)} \rangle| ,$$

for given n, m.

2.1.4 Maximal synchronization index

The function `co_maxsync` computes $\max(\gamma_{n,m})$ in a given range of n, m. It returns the matrix $gamma_{n,m}$, the maximal value of the synchronization index and the corresponding values n and m.

2.1.5 Directionality index I

The function `co_dirin` computes the directionality index

$$d = \frac{c_2 - c_1}{c_2 + c_1} , \tag{4}$$

where

$$c_{1,2} = \frac{\mathcal{N}^{(1,2)}}{\omega_{1,2}} . \tag{5}$$

Here $\mathcal{N}^{(1,2)}$ are norms of the coupling functions $Q^{(1,2)}$. Defined in this way, coefficients $c_{1,2}$ provide an observable-independent measure of directionality.

The directionality index varies from $d = -1$, if system 2 drives system 1, to $d = 1$, if the unidirectional driving is in the other direction; the values $-1 < d < 1$ correspond to a bidirectional coupling.

2.1.6 Directionality index II

The function `co_dirpar` computes the directionality index according to [2], i.e. the coefficients in Eq. (4) are obtained as

$$c_{1,2}^2 = \frac{1}{4\pi^2} \int_0^{2\pi} \int_0^{2\pi} \left(\frac{\partial q^{(1,2)}}{\partial \phi_{2,1}} \right)^2 d\phi_1 d\phi_2 ,$$

2.2 Univariate analysis

2.2.1 Fourier-based $\theta \to \phi$ transformation

Univariate transformation $\theta \to \phi$, based on the expansion of the probability density into Fourier series, is performed by the function `co_fbtransf1`. For its implementation we re-write Eq. (16) from [7] in a form, more convenient for computation:

$$\phi = \theta + \sum_{n \neq 0} \frac{S_n}{in} (e^{in\theta} - 1) = \theta + \sum_{n=1}^{\infty} \left[\frac{S_n}{in} (e^{in\theta} - 1) - \frac{S_{-n}}{in} (e^{-in\theta} - 1) \right] .$$

Using $S_{-n} = S_n^*$ we obtain

$$\phi = \theta + 2\sum_{n=1}^{\infty} \text{Re}\left[\frac{S_n}{in}(e^{in\theta} - 1)\right] = \theta + 2\sum_{n=1}^{\infty} \text{Im}\left[\frac{S_n}{n}(e^{in\theta} - 1)\right] .$$

In addition, to what has been done in [7], Gaussian kernel smoothing is introduced:

$$\phi = \theta + 2\sum_{n=1}^{\infty} \text{Re}\left[\frac{S_n}{in}(e^{in\theta} - 1)\right] = \theta + 2\sum_{n=1}^{\infty} e^{-n^2\alpha^2/2}\text{Im}\left[\frac{S_n}{n}(e^{in\theta} - 1)\right] ,$$

where α is the smoothing parameter.

The function optionally provides the transformation function σ, computed according to

$$\sigma = \frac{d\phi}{d\theta} = 1 + 2\sum_{n=1}^{\infty} e^{-n^2\alpha^2/2}\text{Re}\left(S_n e^{in\theta}\right)$$

on a grid of size `ngrid`.

2.3 Bivariate analysis

The aim of this analysis is to transform simultaneously two protophases to true phases by means of the transformation functions

$$\sigma_1 = \frac{d\phi_1}{d\theta_1} , \qquad \sigma_2 = \frac{d\phi_2}{d\theta_2} ,$$

and to reconstruct Eqs. (3).

An important note is in order. While reconstructing the coupling functions, we obtain some constant terms. Strictly speaking, these terms can differ from the natural frequencies $\omega_{1,2}$ of oscillators, because the coupling functions generally contain a constant term $\hat{\omega}_{1,2}$ and what is recovered by the analysis is the sum $\omega_{1,2} + \hat{\omega}_{1,2}$. If we have only one observation of the coupled system, then we cannot separate these constants. (It can be done if several observations of the system at different coupling strength are available, see [7].) In the following we simply say "frequencies" and denote them as $\omega_{1,2}$ (`omega1` and `omega2` in the code), although strictly speaking these are the sums $\omega_{1,2} + \tilde{\omega}_{1,2}$.

If you do not want to go into details, you can simply use one of two high-level functions `co_fbtransf2` and `co_ittransf2`. These functions consist of the building blocks, also described below; you can use these blocks in order to optimize your particular application. Note that `co_fbtransf2` uses the optimization matlab toolbox; if your distribution does not contain this toolbox, use the function `co_ittransf2`.

2.3.1 High-level function which does everything, Fourier-based technique

The function `co_fbtransf2` performs the whole analysis. Taking two protophases as the input, it performs the transformation and returns frequencies $\omega_{1,2}$ and coupling functions $Q^{(1,2)}$, computed on a grid. Furthermore, it returns the Fourier coefficients of the coupling functions, norms of these functions, true phases $\phi_{1,2}$, and directionality index according to Eqs. (4,5). In order to perform the bivariate $\theta \to \phi$ transformation, this function solves a nonlinear equation system (see Appendix in [7]).

2.3.2 High-level function which does everything, iteration technique

The function `co_ittransf2` does the same as the previous one (it does not compute the Fourier coefficients of coupling functions). The difference is that it uses iteration technique for the solution of the nonlinear equation system (see Appendix in [7]).

2.3.3 Coupling functions for two oscillators: Fourier fit

This computation is performed by the functions `co_fexp1` and `co_fexp2`. Given two (proto)phases the first one provides the coupling function $F^{(1)}(\theta_1, \theta_2)$ by fitting a double Fourier series of the given order. It corresponds to the algorithm, described in [7]. The second function computes both coupling functions $F^{(1)}(\theta_1, \theta_2)$ and $F^{(2)}(\theta_2, \theta_1)$ simultaneously. The output are the Fourier coefficients of the function of protophases $F^{(1)}(\theta_1, \theta_2)$ and $F^{(2)}(\theta_2, \theta_1)$. Optionally, the `co_fexp1` and `co_fexp2` compute $F^{(1)}(\theta_1, \theta_2)$ and $F^{(2)}(\theta_2, \theta_1)$ on a grid.

Note that using `co_fexp2` is faster than using `co_fexp1` twice. Note also, that $F^{(1)}(\theta_1, \theta_2)$ and $F^{(2)}(\theta_2, \theta_1)$ are not the true functions; the latter are obtained by the transformation

$$F^{(1)}(\theta_1, \theta_2), F^{(2)}(\theta_2, \theta_1) \to Q^{(1)}(\phi_1, \phi_2), Q^{(2)}(\phi_2, \phi_1) \ .$$

This transformation is performed by the functions, described below.

2.3.4 Functions for the Fourier-based technique

I. Function `co_fbsolv` solves the nonlinear equation system using the nonlinear equation solver of MATLAB optimization toolbox and returns Fourier coefficients of the transformation functions `sigma1` and `sigma2`. Optionally, it also returns the functions themselves, computed on a grid.

II. Function `co_fbth2phi` performs the $\theta_{1,2} \to \phi_{1,2}$ transformation, using the output of `co_fbsolv`.

III. Function `co_fbnorm` computes the norm of the coupling function, given in terms of its Fourier coefficients. Function `co_fbnorm` works with the output of `co_fexp2`.

2.3.5 Functions for the iteration technique

I. Function `co_itersolv` solves the equation system by iterations and computes the bivariate $\theta_{1,2} \to \phi_{1,2}$ transformation functions $\sigma_{1,2}$. `co_itersolv` works with functions, computed on the grid; it uses the output of `co_fexp2`. It also returns frequencies, true coupling functions, and their norms.

II. Function `co_gth2phi` performs the $\theta_{1,2} \to \phi_{1,2}$ transformation, using the output of `co_itersolv`.

2.4 Additional functions

2.4.1 Plotting functions

I. Function `co_plotcplf` plots the computed coupling function. Function `co_plot2cplf` plots two coupling functions in the same graphic window.

II. Functions `co_plotcoef` and `co_plot2coef` plot color-coded Fourier coefficients of one or two coupling functions, respectively.

2.4.2 Correlation of coupling functions

Toolbox functions `co_fbcfcor` and `co_gcfcor` compare two coupling functions, computing their correlation. `co_fbcfcor` operates with Fourier coefficient, whereas `co_gcfcor` works with functions, given on a grid.

3 Examples

The toolbox function are illustrated by several examples with artificially generated data. For each example we provide the matlab code, the data file, and the pdf file with the output and comments. These files can be downloaded from the toolbox web-page. More examples are coming in the nearest future.

3.1 Two coupled van der Pol oscillators

The model is

$$\begin{aligned} \ddot{x}_1 - \mu(1-x_1^2)\dot{x}_1 + \omega_1^2 x &= \varepsilon_1(\dot{x}_2 - \dot{x}_1) \ , \\ \ddot{x}_2 - \mu(1-x_2^2)\dot{x}_2 + \omega_2^2 x &= \varepsilon_2(\dot{x}_1 - \dot{x}_2) \ , \end{aligned} \tag{6}$$

where the parameters are $\mu = 0.5$, $\omega_1 = 1.11$, and $\omega_2 = 0.89$.

First two examples are implemented by the functions `co_example1` and `co_example2`. For the input these functions use the data file `co_vdp2.mat`. This data set corresponds to symmetrically coupled van der Pol oscillators with $\varepsilon_1 = \varepsilon_2 = zz$. Functions `co_example1` and `co_example2` illustrate the Fourier-based algorithm. The first example program `co_example1` is based on the high-level function `co_fbtransf2`. If you do not go into details, you may take this file and modify it to read your data. The second example program `co_example2` plots all intermediate results and illustrates the whole procedure step-by-step. Note that you need matlab optimization toolbox to run these examples.

Third and forth examples are implemented by the functions `co_example3` and `co_example4`. They illustrate the iteration technique. Again, the first of these two is based on the high-level function `co_ittransf2`, while the second performs and illustrates every step separately. These examples take the data from `co_vdp2uni.mat`; this data set corresponds to unidirectionally coupled oscillators with $\varepsilon_1 = zz$ and $\varepsilon_2 = zz$. You can easily modify these files to read and process

References

[1] M. G. Rosenblum, A. S. Pikovsky, J. Kurths, C. Schäfer, and P. A. Tass. Phase synchronization: From theory to data analysis. In F. Moss and S. Gielen, editors, *Neuro-informatics and Neural Modeling*, volume 4 of *Handbook of Biological Physics*, pages 279–321. Elsevier, 2001.

[2] M. G. Rosenblum and A. S. Pikovsky. Detecting direction of coupling in interacting oscillators. *Phys. Rev. E*, 64(10):045202, 2001.

[3] A. Pikovsky, M. Rosenblum, and J. Kurths. *Synchronization. A Universal Concept in Nonlinear Sciences.* Cambridge University Press, Cambridge, 2001.

[4] M. G. Rosenblum, L. Cimponeriu, A. Bezerianos, A. Patzak, and R. Mrowka. Identification of coupling direction: Application to cardiorespiratory interaction. *Phys. Rev. E*, 65(4):041909, 2002.

[5] M. G. Rosenblum, L. Cimponeriu, and A. S. Pikovsky. Coupled oscillators approach in analysis of bivariate data. In B. Schelter M. Winterhalder and J. Timmer, editors, *Handbook of Time Series Analysis*, pages 159–180. Wiley-VCH, Weinheim, 2006.

[6] B. Kralemann, L. Cimponeriu, M. Rosenblum, A. Pikovsky, and R. Mrowka. Uncovering interaction of coupled oscillators from data. *Phys. Rev. E*, 76:055201, 2007.

[7] B. Kralemann, L. Cimponeriu, M. Rosenblum, A. Pikovsky, and R. Mrowka. Phase dynamics of coupled oscillators reconstructed from data. *Phys. Rev. E*, 77:066205, 2008.

[8] Y. Kuramoto. *Chemical Oscillations, Waves and Turbulence.* Springer, Berlin, 1984.